MoHiD: A Scalable Mobility Platform based on Hierarchical Distributed Hash Table

Haeun Kim, Jaehyun Park, Sangyup Han, and Myungchul Kim
School of Computing, KAIST, Daejeon, Republic of Korea
{hukim, jae519, ilu8318, mck}@kaist.ac.kr

Abstract

In order to resolve the scalability and handover performance issues of existing approaches, we propose a scalable mobility platform which is based on the hierarchical Distributed Hash Table (HDHT), referred to here as MoHiD, to provide host mobility through a DHT. In MoHiD, the location information of the hosts is stored in the HDHT running on the MoHiD Access Router to provide mobility. In the HDHT, the storage level of each entry can be specified, which drastically reduces the handover delay by limiting the number of overlay hops occurring during the query processing steps. In order to demonstrate the feasibility of MoHiD, we used a commercial cloud to measure the entry query time of the HDHT on the global scale and constructed a testbed to measure the handover performance. The experimental results show that the HDHT query delay and the total handover delay are 16.7 ms and 115.9 ms, respectively, this providing evidence of rapid handovers.

1. Introduction

At present, there are very high numbers of mobile nodes connected to networks. In addition to current mobile devices such as smartphones and laptops, in the near future, most mobile objects around us, such as sensors, glasses, and cars, are also expected to be connected to the Internet. According to Cisco, more than 50 billion devices will be connected to the Internet by 2020 [1]. Furthermore, Gartner predicts that more than a quarter of the world's vehicles will be connected to the Internet by 2020 [2], and Machina Research predicts that by 2024, the number of machine-to-machine connections will exceed 27 billion [3]. These statistics strongly suggest that Internet-connected objects with mobility will increase sharply in the future.

As the number of Internet-connected objects with mobility increases, mobility support and management will become more important in the future Internet environment. Therefore, it is critical to design a mobility support platform in preparation for the future. Current networks operate using a centralized scheme in which all user traffic passes through the core network. For example, the 3rd Generation Partnership Project provides network-based IP mobility using the General Packet Radio Service tunneling protocol [4] instead of Proxy Mobile IPv6 (PMIPv6) [5]. This centralized scheme has the advantage of a central anchor operating in a simple manner, rerouting packets to the current location of the mobile node (MN). However, when the anchor becomes inoperable, it causes what is termed the single-point-of-failure problem, after which the entire system cannot operate. Furthermore, scalability issues when the number of users using the system increases can occur [6]. Therefore, in the future Internet environment, a distributed scheme must be developed and applied.

In order to resolve the chronic problems associated with the current centralized scheme, the Internet Engineering Task Force (IETF) is attempting to organize the requirements of the distributed mobility management (DMM) scheme and to introduce existing IP mobility solutions into the data plane of the DMM environment [6-7]. However, this scheme is not currently used due to performance degradation problems such as a complex tunneling method and a suboptimal path [8-9]. In addition, research has begun on a full-DMM system that is distributed not only to data planes but also to control planes. For example, when using the distributed hash table (DHT) as a control plane, there is a fatal disadvantage in that the query delay increases as the number of participating nodes increases [10]. In order to resolve the delay problem of the DHT, various solutions such as a hierarchical DHT configuration [11-13], a one-hop DHT configuration [14], a simultaneous multiple query use [15], and consideration of the physical location of DHT nodes during the node ID assignment process [16] have been proposed. However, because these studies only focused on improving the DHT performance itself, there has been a general lack of consideration of the mobility of the host.

Toward solving the existing mobility management problems, such as scalability and query delay problems,

URI: http://hdl.handle.net/10125/50617
ISBN: 978-0-9981331-1-9

in this paper we propose a scalable mobility platform based on hierarchical DHT, termed MoHiD. MoHiD consists of the hierarchical DHT (HDHT) and a MoHiD access router (MAR) which provides host mobility. The HDHT resolves the delay problem that occurs when a flat DHT is applied on the global scale, and it limits the number of overlay hops required for the query. The information needed to support mobility for MNs is stored in the HDHT and the mobility of each MN is supported by the MAR.

The significant contributions of MoHiD are described below.

1. We propose a mobility platform that supports a full-DMM scheme with a distributed control plane through the HDHT. Owing to its fully distributed design, it resolves problems such as scalability, the single-point-of-failure issue, and the handover performance of existing centralized mobility support schemes.
2. We propose a new HDHT where the number of overlay hops is limited by hierarchically configuring the DHT and specifying the level at which the entry is stored.
3. MNs connecting MoHiD do not need to be modified for mobility support because MoHiD is a type of network-based architecture. Moreover, because the proposed scheme is hierarchical, it can be applied gradually from the edge network, which is interoperable with existing networks.
4. We measured the HDHT query delay on the global scale using a commercial cloud service and measured the performance of the handover by implementing MoHiD in a testbed environment. The result demonstrates that MoHiD is adoptable as a global-scale mobility platform.

In this paper, we introduce related works in Section 2. Section 3 describes the proposed system, and Section 4 describes the experimental procedure and the results of the verification of the proposed system. Then, Section 5 presents the conclusions and suggests future research directions.

2. Related work

In this section, we explain earlier work related to MoHiD.

2.1. Mobility management

The mobility support solutions can be categorized as host-based methods or network-based methods. Host-based methods imply that the MNs must be modified. Mobile IP (MIP) [8] and Mobile IPv6 (MIPv6) [9] are typical techniques. These schemes must modify the MN in order to send the signal information directly when the handover occurs, and they are rarely used due to performance problems such as triangular routing.

Network-based methods only need to modify the network. These methods are classified into the centralized, partially distributed, and fully distributed types [17].

Proxy MIPv6 (PMIPv6) [5] is a centralized method. In PMIPv6, the mobile access gateway (MAG) sends and receives signal messages for a handover on behalf of the MN. The MAG functions as an access router, creates a bi-directional IP tunnel with the local mobility anchor (LMA), and sends and receives packets destined for the MN. One LMA is connected to several MAGs, and all packets related to the mobility process pass through the LMA. Therefore, if an error occurs in the LMA, the communication of all nodes that need to pass through the LMA can be disconnected. If there are large numbers of MNs, scalability problems are likely to occur.

The DMM method was proposed to solve the scalability problem, which is a chronic problem related to centralized methods. DMM methods can be classified as partially distributed methods and fully distributed methods. In both schemes, the data planes are distributed, with the only difference being whether or not the control plane is distributed. The partially distributed scheme can be introduced based on PMIPv6 [18] or software-defined networking; these use a centralized database known as a control mobility database and a centralized network controller, respectively.

MoHiD as proposed in this paper is a type of fully distributed DMM scheme because the HDHT is used on the distributed control plane. It is also a network-based method because it requires modification of the network only without requiring any modification of the MN.

2.2. Distributed Hash Table (DHT)

The most widely used and best known DHT systems are Kademlia [15], Chord [19], and the Content Addressable Network (CAN) scheme [20]. In the DHT, the entries are distributed and stored among the participating nodes. The nodes participating in the DHT have an arbitrary node ID, and the key value of the DHT entry is generated using a hash function. When the node participating in the DHT sends a query, the query is forwarded to the node responsible for the entry. The node IDs assigned to the DHT nodes are assigned arbitrary values in the flat namespace without considering the physical distance. Therefore, even if the logical distance between node IDs is short, the physical distance can be great. For this reason, even if the desired entry is physically close, the query can be transmitted to a remote place, which causes a lengthy query delay [16]. If the DHT nodes are globally distributed, the delay increases due to this problem, resulting in a long query delay [21]. Therefore, a flat DHT is not suitable for use with a full-DMM system.

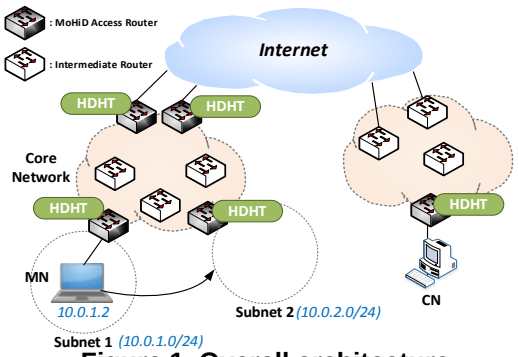

Figure 1. Overall architecture

In order to solve the delay problem that arises when the DHT nodes are distributed globally, various solutions have been proposed, such as a hierarchical DHT configuration [11-13], a one-hop DHT configuration [14], the scheme known as simultaneous multiple query use [15], and schemes which consider the physical location. In particular, in one of these studies [12], when the DHT is hierarchically structured, it was demonstrated that the routing speed was most efficient when the level was set to '2' or '3'. However, because these methods were originally proposed to reduce the DHT query delay, the degree of consideration of the host mobility is insufficient.

2.3. Name Resolution System (NRS)

The NRS is a system that provides information about the current location of the host based on the host ID. Example systems include LISP [22], XIA [23], and Mobility-First [24]. For scalability purposes, the NRS, similar to our approach, can be distributed based on the DHT; examples of these systems include DHT-MAP [25], Mobility First's DMap [26], LMChord [16], MDHT [27], and CoDoNS [28].

DHT-MAP [25] was designed based on CAN [20]. This results in a neighbor node functioning as a backup node, and a joining rule and leave process are also introduced. These features enable CAN to function well as the NRS. However, because the resolution delay exceeds 480 ms given that the physical distance between the nodes is not considered, this scheme can be challenging if used where seamless mobility is required.

DMap [26] applies multiple hash functions to a globally unique identifier (GUID) and finds the IP addresses of the nodes with the desired entry at a time. However, due to the method of randomly distributing entries to several nodes, the physical distance is not considered and the delay can be long. In addition, because this scheme assumes that all ASs that exist in the network participate in the NRS, it can be applied to the future Internet environment. Furthermore, the MN must directly register the location information and perform the query. Therefore, if the MN performs the handover during the communication step, the correspondent node (CN) must query the location of the MN again. Only experiments on the query delay have been conducted, whereas experiments on the handover delay have not been undertaken.

LMChord [16] solved the mismatch problem between physical and overlay networks by applying the Markov decision process to the DHT. This involves two-level DHTs. At the higher level, the DHT relays messages between the lowest DHTs. This causes a scalability problem in that the DHT at higher level must maintain mapping information pertaining to all MNs on the network. In addition, experiments on the handover of the MN have not been conducted.

MDHT [27] is a global NRS that can be used in information centric networking by hierarchically configuring the DHT. However, when the entry is stored in the MDHT, the pointer information is stored in all DHTs existing between the highest DHT and the lowest DHT. Furthermore, when the entry is stored in the MDHT, the information is always propagated to the upper NRS tree. As a result, the top DHT must maintain the entries for all IDs

In summary, we propose the MoHiD architecture, which, unlike in previous studies, considers the relationship between the mobility of the host and the DHT. In addition, we measured the query delay on the global scale while also measuring the handover delay through handover experiments in a testbed environment and then analyzing the effect of the HDHT query delay with respect to the handover delay.

3. Proposed MoHiD architecture

In this section, we describe the HDHT system, in which the information necessary for host mobility is stored. The role of the MAR in supporting the mobility of the host based on the information stored in the HDHT is then described. Finally, the process in which the handover of the MN is supported by the processing performed by the MAR with the information stored in the HDHT is explained.

3.1. Overall architecture

The overall behavior of MoHiD is depicted in Figure 1. The MARs that provide network access to the MNs store the information in the HDHT that is necessary for MN mobility. Using the information stored in the HDHT, the MAR delivers the packet normally to the MN, even if the MN changes its network location. For example, if the MN that is communicating with the CN moves from Subnet 1 to Subnet 2, the communication between the MN and CN can continue in this case. The detailed operation procedure is covered in Section 3.4. Please note that all legacy routers can be used as they are except for the MARs, as packet routing between MARs operates in the typical manner. In addition, because the mobility of the host is supported by the MAR, modification of the host is unnecessary.

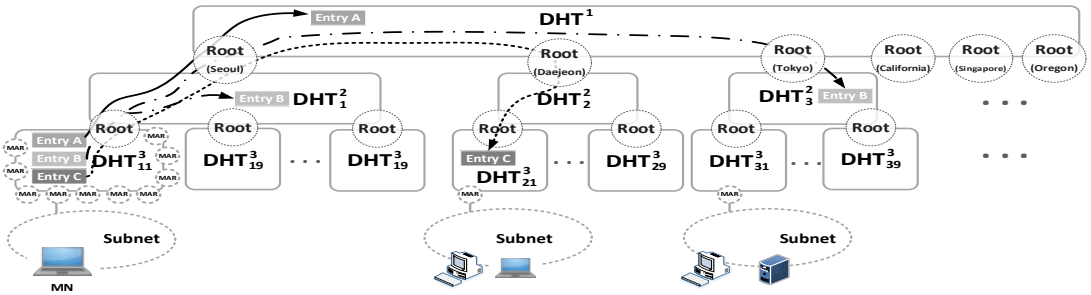

Figure 2. Registration process

3.2. HDHT system

The HDHT system proposed in this paper has a hierarchical structure with a total of three levels, as depicted in Figure 2. The number of levels can be adjusted according to the network size. When constructing the DHT hierarchically, the nodes participating in the lowest DHT constitute the group of physically close MARs. For example, MARs belonging to a building or a campus can form the lowest DHT group. Because the physical distance between the nodes in the same DHT is close, the query for the DHT at the lowest level is completed very quickly. In this paper, each DHT participating in the HDHT is labeled as DHT_j^i. The superscript i indicates the DHT level, and j in subscript indicates the ID of the DHT. The actual ID of the DHT is allocated randomly; however, for convenience, it is represented using consecutive numbers in this paper. The entries stored in the DHT contain the information necessary to ensure host mobility.

The HDHT application running on the MAR participates as a node in the DHT. The DHT when operating consists of a one-hop DHT. In the one-hop DHT, each MAR maintains a DHT routing table that contains information about all MARs participating in the same DHT. Using this routing table, the MAR instantly identifies another MAR with the desired entry. The root node is the first node participating in the DHT; it is a special node that undertakes the registration and inquiry message transfers between the DHT at the upper level and that at the lower level. It is also responsible for propagating the latest routing table for the MARs participating in the DHT.

The DHT routing table contains the node IDs, IP addresses, and port numbers of all MARs participating in the same DHT. The HDHT uses a SHA-256 hash function [29] with a 32-byte length. However, the hash function can be replaced with another hash function when required. All MARs participating in the same DHT have the same DHT routing table. In order to achieve this, when a new MAR participates in the DHT, it sends a bootstrap message to the root node, which adds the node information to the routing table and propagates it to the participating MARs. Furthermore, when the MAR participating in the DHT leaves, the root node propagates this event to the other MARs.

When determining the MAR responsible for the entry, the logical distance between the key value of the desired entry and the node ID stored in the routing table is calculated and the MAR is determined as the entity having the smallest value. In this paper, the XOR operation is used for the logical distance calculation in the DHT because XOR metric distances have the property of triangle inequality [15].

As noted above, when inserting an entry into the HDHT, we can choose the HDHT level at which the entry is stored. The HDHT level is determined based on the node mobility range. The process of registering an entry in the HDHT presented in Figure 2 is described by the following pseudocode.

```
▷R : DHT routing table of a MAR
▷K : an entry key
▷E : an entry
▷L : a specified level of entry
▷MyLevel : own level of a DHT node
procedure REGISTER_OR_UPDATE_ENTRY_IN_HDHT(K, E, L)
① MAR ← Decide_Responsible_MAR(R, K)
② Insert_Entry(K, E, MAR.IP, MAR.Port)
④ Registration_Up_Forward_To_Root(K, E, L)

▷S : DHT storage of the MAR
procedure INSERT_REQUEST_ARRIVED(K, E)
③ Add_Entry(S, E)

▷RH : routing table of a higher level DHT
procedure UP_FORWARD_TO_ROOT_ARRIVED(K, E, L)
   MAR ← Decide_Responsible_MAR(RH, K)
⑤ Registration_Up_Forward(K, E, L, MAR.IP, MAR.Port)

▷RL : routing table of a lower level DHT
procedure UP_FORWARD_ARRIVED(K, E, L)
   if MyLevel == L then
⑦    Add_Entry(S, E)
   if (MyLevel == 1) and (L == 1) then
      MAR ← Decide_Responsible_MAR(RL, K)
⑧    Registration_Down_Forward(K, E, L, MAR.IP, MAR.Port)
   else
⑥    Registration_Up_Forward(K, E, L)

procedure DOWN_FORWARD_ARRIVED(K, E, L)
   if MyLevel == L then
⑨    Add_Entry(S, E)
   if (MyLevel != 3) and (L != 3) then
      MAR ← Decide_Responsible_MAR(RL, K)
⑩    Registration_Down_Forward(K, E, L, MAR.IP, MAR.Port)
```

Hierarchical DHT Entries

Key	MN_IP	MAR_IP	MAR_LIST	Level
0x6000...	10.0.1.2	10.0.1.1	10.0.1.1, 10.0.16.1	3
0x534E...	10.0.16.32	10.0.16.1	10.0.16.1, 10.0.48.1	1
0xE4EA...	10.0.96.128	10.0.48.1	10.0.96.1, 10.0.48.1, 10.32.0.1, ...	2

Figure 3. HDHT entries

The entry is always stored in the lowest DHT to which the MAR requesting the entry registration belongs. The MAR uses the routing table of the lowest DHT to which it belongs and determines the MAR responsible for this entry (①) and sends a message to register the entry in the MAR (②). The MAR that receives this message registers the entry in its own storage (③). In addition, the MAR requesting the entry registration also sends a message requesting that the entry be forwarded to the upper DHT to the root node (④). The information of the root node is maintained from the time the MAR participates in the DHT. Because the root node also has the routing table of the upper DHT, it can forward the request to the upper DHT (⑤). The registration request is consecutively forwarded to the upper DHT (⑥), and the entry is stored when it passes the requested level during the forwarding process (⑦). When the registration request message arrives at the top DHT, the other top DHT node to store the entry is determined through a logical distance calculation, and the forwarding direction is changed to the downward direction (⑧). When the MAR receives the registration message that has changed to the downward direction, it stores the entry when the DHT level in which it participates is equal to the requested level (⑨). If the level is different, the registration message is transmitted to the lower level (⑩). When the entry is updated, the entry update message is also transmitted in the manner utilized during the registration process.

In Figure 2, entry A corresponds to DHT_{11}^{3} requesting that the registration with the HDHT level be set to '1'. Entry A is stored first in DHT_{11}^{3}, where the request originated, and is forwarded to the top DHT. When it is delivered to the top DHT, the entry is stored at DHT^{1} because the requested level and the level of the top DHT are identical. Entry B is where a registration request is made by setting the HDHT level to '2' in DHT_{11}^{3}. In this case, the entry is also stored in DHT_{11}^{3}, which initiated the request, and the entry is routed to the top DHT. Because the level of DHT_{11}^{3} is identical to the requested level of the entry during an upward propagation, the entry is stored there. After passing through the top DHT, the direction of the propagation changes to the downward direction and the entry is finally stored in DHT_{3}^{2}. In this case, because there is no need to deliver to the DHT operating at level '3', no additional propagation is made. Finally, entry C corresponds to when a registration is requested while designating an HDHT level of '3'. The entry is stored in DHT_{11}^{3}, in which the MAR is engaging, and then forwarded to the top DHT. The forwarding direction is changed to the downward direction after reaching the top DHT. Finally, it is transferred to DHT_{21}^{3} and the entry is stored. As illustrated by the above process, because the entries are not stored in the top DHT (except when the entry level is '1'), there is an advantage in that the top DHT does not need to store all entries.

The process of sending an entry query message is identical to that when the entry is registered. The query direction always begins in the upper direction and changes to the lower direction after passing through the top DHT. Each time a query is delivered, the number of overlay hops increases by one.

The number of overlay hops and the query delay vary greatly depending on the HDHT level where the entry is stored and the location of the MAR querying it. For example, for entry C, the MARs belonging to the same DHT_{11}^{3} can complete the query within a single overlay hop. However, if the MAR participating in DHT_{12}^{3} queries entry C, the query is forwarded to DHT_{21}^{3} through the top DHT. In this case, although the desired entry is physically located close, i.e., DHT_{11}^{3}, inefficiency arises because the query goes through the top DHT. If the entry C level is set to '2', we can fetch the desired entry in three overlay hops because the entry is stored in DHT_{1}^{2} despite the fact that the query begins at DHT_{12}^{3}. Therefore, when registering an entry in the HDHT, it is necessary to determine the level to be stored considering the location of the MARs that are likely to query the entry.

3.3. MoHiD Access Router (MAR)

A typical access router assigns an IP address to each of its connected hosts and functions as a default gateway. In this paper, even if the MN moves to a network composed of other subnets, the IP address of the MN is maintained.

After the allocation of the IP address, the key value of the entry can be generated by applying a hash function to the IP address assigned to the host. The MAR can verify that the information of the host is stored in the HDHT by sending a query. If such an entry exists, it signifies that the host has already connected to the network. In this case, the mobility information of the MN is updated in the entry. If such an entry does not exist, this indicates that the host is connecting to the network for the first time. Therefore, a new HDHT entry containing the location information of the MN is created and inserted. The contents of such an entry are described in Figure 3. The key of the entry is the hash value of the IP address of MN, and the values of the entry are MN_IP, MAR_IP, MAR_LIST and the level at which the entry is inserted.

The following pseudocode describes the tasks performed by the MAR to register or update the location information of the MN.

```
▷C : a cache in an MAR
procedure A_NEW_MN_IS_ASSOCIATED(IP)
① K ← Generate_Key(IP) ▷K : a key
② E ← Retrieve_Entry(K) ▷K : an entry
   if Is_None(E) then
```

```
③     E ← Allocate_New_Entry()
      E.MN_IP ← IP
      E.Level ← 3
④ E.MAR_IP ← Get_IP_Of_This_Switch()
⑤ E.MAR_LIST ← E.MAR_LIST + E.MAR_IP
⑥ Register_Or_Update_Entry_In_HDHT(K, E, E.Level)
⑦ C ← C + E
   M ← (K, E.MAR_IP) ▷M : cache update message
   for all S ∈ (E.MAR_LIST − E.MAR_IP) do
      Send_A_Cache_Update_Message(S, M)
```

First, the key value of the entry is generated based on the IP address of the MN (①). The MAR sends a query using the key value of the entry to verify whether the entry has been registered (②). If the corresponding entry does not exist, a new entry is created and the IP address of the MN currently connected is recorded in the MN_IP field, with the HDHT level at which the entry will be stored specified at this time (③). The default value of the level at which the entry is stored is '3', which can be modified later. After replacing the MAR_IP field with its own IP address (④), the MAR also adds its own IP address to the MAR_LIST field (⑤). The MAR then requests the registration of the created or modified entry information from the HDHT (⑥). At this time, the level at which the entry is inserted must be specified. For the above pseudocode, the HDHT level is not modified when the entry is updated. After storing the entry information in its own cache (⑦), it sends a cache update message to the MARs that have cached the entry (⑧).

The MARs that receive the cache update message check their cache for an entry matching the key. If there is a matching entry, the cache entry is updated. This is expressed by the following pseudocode.

```
▷C : a cache in MAR
procedure CACHE_UPDATE_ARRIVED(K, MAR_IP)
   E ← Retrieve_Cache_Entry(K) ▷E : a cache entry
   E.MAR_IP ← MAR_IP
```

Each time the MN moves through the two functions, i.e., A_NEW_MN_IS_ASSOCIATED() and CACHE_UPDATE_ARRIVED(), the location information is updated in the HDHT and the cached entries in the MARs are updated.

The packet processing steps performed by the MARs using the information stored in the HDHT are described in the following pseudocode.

```
▷C : a cache in MAR
▷P : a packet from the associated host
procedure PACKET_PROCESSING(P)
   IP ← Obtain_IP_From_Packet(P)
① K ← Generate_Key(IP) ▷K : a key
② E ← Retrieve_Cache_Entry(K) ▷E : a cache entry
③ if P does not have optional header then
④     if Is_None(E) then
         Packet_Forwarding(P)
         Query_To_HDHT(K)
         return
⑤     P.dst_IP ← E.MAR_IP
      Append_Optional_Header(P, K)
      Packet_Forwarding(P)
   else
⑥     if Is_None(E) then
         Drop_The_Packet(P)
         return
⑦     P.dst_IP ← E.MN_IP
      Remove_Optional_Header(P)
      Packet_Forwarding(P)
```

When a packet arrives, the MAR initially applies a hash function to the IP address of the destination node in order to generate a key value, after which it verifies whether there is a cached entry in the cache (①, ②). It also verifies whether there is an optional header added by the other MAR in the packet (③). If there is no optional header, steps ④ and ⑤ are performed. If there is no cached entry, it immediately forwards the packet and sends a query to the HDHT (④). If the location information of the destination node is cached, the destination IP address of the packet is modified to MAR_IP (⑤), which is the IP address of the MAR to which the destination node is connected, and the packet is forwarded after adding the key value of the entry to the option header. In contrast, if a packet with an optional header added by the other MAR is received and there is no cached entry, the packet is dropped (⑥). If a cached entry exists, the destination IP address of the packet is restored to the IP address of the host (⑦), after which the optional header is removed and the packet is forwarded. In this case, the packet is normally delivered to the MN. Because the MAR performs the PACKET_PROCESSING() function, even if the MN moves to another MAR, the packets destined for the IP address of the MN are delivered to the MAR to which the MN is newly connected. When the packets arrive at the MAR to which the MN is connected, the destination IP address is restored to the IP address of the MN so that the packets will arrive normally at the MN. For example, suppose that the MN is connected to MAR A and that the CN is connected to MAR C, as depicted in Figure 4. When the CN sends a packet to the MN, MAR C verifies whether this MN information is stored in its cache. If this is the case, MAR C modifies the destination IP address by changing it to the IP address of MAR A and inserts the key value of the entry into the optional header. When this packet arrives at MAR A, MAR A restores the destination IP address to the IP address of the MN, removes the optional header, and forwards the packet to the MN.

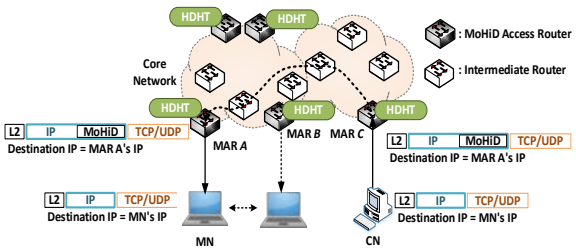

Figure 4. Packet processing at the MAR

The procedure for handover processing during communication is discussed in Section 3.4.

The MAR is implemented by modifying the Edge Switch (ES), the process of which is described in earlier work [30]. The ES is implemented by modifying the source code of the Open vSwitch (OVS) kernel module [31].

3.4. End-host handover process

Figure 5 presents a sequence diagram of the handover support process when the MN connects to a new MAR. First, the MN establishes an L2 association with the new MAR (①-②). The MAR assigns an IP address to the MN through the DHCP (③-⑥). During this process, because the IP address of the MN is known, the key value of the entry can be generated. The key is used to query the HDHT entry for the MN (⑦). If there is an existing entry, the MAR updates it; otherwise, it registers a new entry (⑨). When updating an entry, it sends a cache update message to the MARs included in MAR_LIST (⑩). The MARs that receive the cache update message update their cache entries. That is, the A_New_MN_Is_Associated() function is executed in the new MAR, and the Cache_Update_Arrived() function is executed in the MARs included in MAR_LIST.

The communication of the MN is restored when the packets are transferred to the new MN location after the entries caching the location information of the MN are updated. The components of the handover latency are described in detail below.

$$T_{Handover\_Delay} = T_{L2\_Association} + T_{DHCP\_Process} + T_{HDHT\_Query} + T_{CacheUpdateMessage} + T_{CacheUpdate} + RTT_{CN\text{-}MN}/2$$

$T_{L2\_Association}$ is the time required for the L2 association when the MN moves to the new MAR. $T_{DHCP\_Process}$ is the time taken when allocating an IP address to the MN through the DHCP. T_{HDHT_Query} is the time required to query and fetch the HDHT entry for the MN. Through this query, the new MAR can recognize the MAR_LIST field. $T_{CacheUpdateMessage}$ is the time taken for the cache update messages to be delivered to the MARs included in MAR_LIST. $T_{CacheUpdate}$ is the time required for a MAR that receives a cache update message to modify its cache entry internally. Finally, $RTT_{CN\text{–}MN}/2$ is the time required for the packet sent by the CN to be delivered to the MN.

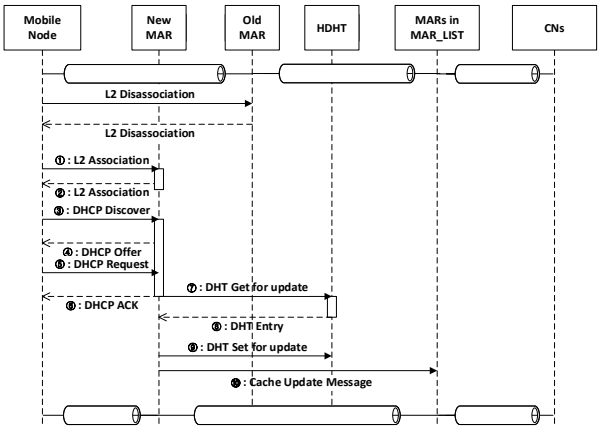

Figure 5. Handover process

The time required for the cache update message to propagate is nearly equal to $RTT_{CN\text{–}MN}/2$, and the time required to update the cache entry internally is sufficiently small so as to be negligible. Therefore, the components of the handover latency are briefly described, as follows:

$$T_{Handover\_Delay} \approx T_{L2\_Association} + T_{DHCP\_Process} + T_{HDHT\_Query} + RTT_{CN\text{-}MN}$$

The values of $T_{L2\_Association}$ and $T_{DHCP\_Process}$ are measured through experiments and are considered to be fixed values. $RTT_{CN\text{–}MN}$ depends on the location of the CN, with the outcome being different recovery delays for each session depending on $RTT_{CN\text{–}MN}$. The focus of this paper is on the HDHT delay, which is indicated as T_{HDHT_Query}. As mentioned above, the query delay can vary significantly depending on the HDHT level of the entry and the location of the MAR requesting the entry.

4. Experiments and evaluation

We used Mininet [32] and Amazon Elastic Compute (EC2) [33] to measure the performance of MoHiD. As depicted in Figure 2, we configured a HDHT topology consisting of physically distant locations, with EC2 hosts positioned in the geographical regions of California, Oregon, Singapore, Tokyo, and Seoul, and with the lab server in Daejeon, Korea participating as the top DHT. Within the EC2 host and the lab server, we executed Mininet and created 100 virtual hosts internally. We created the HDHT as depicted in Figure 2 by allowing the virtual hosts to participate in the HDHT.

The following evaluation scenarios were used.

- Query Delay: The node participating in the lowest DHT operating in Seoul undertook entry registration for 150,000 entries, with 50,000 entries per level. Subsequently, we measured the query delay at the lowest DHT node in another region.
- Handover Delay: We measured the handover delay of the MN in the MoHiD testbed implementing

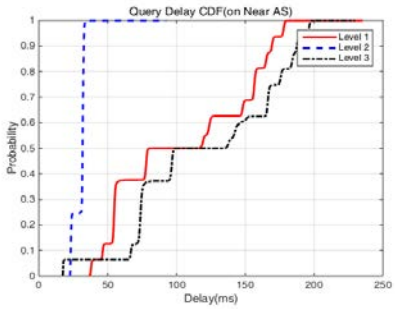

Figure 6. CDF for the query delay on Near AS according to the HDHT level

the HDHT and MAR. We measured the delay of the communication recovered from the disconnection to the recovery when the MN performed a handover during the communication using the iPerf application [34].

4.1. HDHT topology

There were six nodes participating in the top DHT, including the EC2 hosts and the lab server, as shown in Figure 2. The DHT running at level '2' consisted of one root node that delivers messages to the top DHT and nine member nodes. These member nodes operate as root nodes for the DHT nodes operating at level '3'. Finally, the DHT running at level '3' consisted of one root node in the form of a regular DHT node at level '2' and ten member nodes. For example, the EC2 host operating in Tokyo included one node participating in the top DHT; one DHT operating at level '2', indicated here as DHT_3^2; and nine DHTs running at level '3', denoted here as DHT_{31}^3 to DHT_{39}^3. The MARs participate in the DHT operating at level '3' as member nodes.

4.2. Results of the query delay

In MoHiD, the MAR queries the HDHT when performing packet processing. Because the query delay significantly affects the handover delay, we conducted an experiment to measure this. In order to ensure a clear understanding of the experimental results, it was assumed that the set of nodes participating in the same lowest DHT constitutes the AS.

For the experiment, DHT_{11}^3, which is the lowest DHT in Seoul, inserted 150,000 entries at a rate of 50,000 entries per level. When the HDHT level was set to '1', the entries were stored in the top DHT. When the HDHT level was '2', the entries were stored in DHT_{11}^3, DHT_1^2, and an arbitrary DHT operating at level '2'. Finally, when the HDHT level was set to '3', the entries were stored in DHT_{11}^3 and dispersed throughout the DHTs at level '3'. We measured the time to completion when the nodes participating in the lowest DHT in each region sent queries.

Table 1. Query Delays

Geographical regions	Level = 1		Level = 2		Level = 3	
	Avg. (ms)	S.D. (ms)	Avg. (ms)	S.D. (ms)	Avg. (ms)	S.D. (ms)
Local AS	16.5	5.4	16.5	5.4	16.6	5.3
Near AS	104.7	50.2	29.9	3.8	121	53.7
California	134.5	68.7	144.2	74.4	150.3	73.8
Daejeon	141.1	90.5	146.8	98.3	159.8	90
Oregon	139.4	79.3	146.4	88	152.5	87.5
Singapore	206.7	94.9	216.4	100.9	225.3	94
Tokyo	104.4	34.1	114	40.4	122.9	34

Table 1 shows the measured query delays from each node in the different geographical regions. The region of "Local AS" (DHT_{11}^3, which is located in Seoul in our experiment) refers to those where a node belongs to the DHT that inserts the entries, and the region of "Near AS" (DHT_{12}^3, which is also located in Seoul) refers to those containing nodes participating in the DHT which is physically closest to the lowest DHT that inserted the entries. The remaining cases are those in which the query is made at the lowest DHT node in each specified area. Note that each average (Avg.) and standard deviation (S.D.) value pertaining to a query delay was calculated as a result of 50,000 runs.

The second and third columns in Table 1 present the query delays when the level of entries is set to '1'. When a query was made in the Local AS that inserted the entry, the query was completed immediately when the node that sent the query owned the entry. In other cases, the query was completed in the top DHT because the required entry is stored only in the top DHT.

The query times measured when the entry level was set to '2' are depicted in the fourth and fifth columns of the Table 1. Likewise, if the query was started in the Local AS, the query was completed in a very short time, and because the entry was stored in the DHT to which it belonged, the query was completed within 16.5 ms on average. The query that started in the Near AS completed the query within an average of 29.9 ms because the lowest DHT to which it belonged had no entry, but the query was completed in the upper DHT. Figure 6 shows the CDF of query delays on the Near AS for entry levels '1', '2' and '3'. All of the query delays are less than 50 ms for entry level '2', while the average query delays are higher than 100 ms for remaining entry levels. This result implies that setting the entry level to '2' can bring a significant advantage in terms of the handover delay. The queries initiated from the lowest DHT nodes in other regions had query times similar to those when the level was 1, and additional delays incurred because the number of overlay hops increased by one additional level.

The query times measured when the entry level was set to '3' are depicted in the sixth and seventh columns of Table 1. In all cases, except for the query that started from the Local AS, the query time was confirmed to be the maximum. Note that using entry level '3' does not consume the storage areas of the upper-level DHTs (i.e., DHT levels 1 and 2). Setting objects, which tend

to have less movement, to level '3' would reduce storage use by the upper DHTs, which could result in a scalable HDHT.

In summary, it is most efficient to designate entry level to '3' in order to minimize the storage consumption in the upper DHT when the MN moves only within the AS. If the MN moves out of the AS but moves within the DHT range operating at level '2', it is better to set the entry level to '2'. If we do not know where the query for the entry will occur, it is better to set the level to '1', where the maximum number of overlay hops is always guaranteed to be the lowest. It is also recommended to set the level to '3' if a query is unlikely to occur or is not sensitive to delays.

4.3. Handover testbed

The testbed topology for measuring the handover delay of the MN is constructed based on Figure 4. There are three MARs by which Host 1 is connected to MAR A via WLAN, and Host 2 is connected to MAR C via a wired network. Host 1 and Host 2 use the iPerf application to communicate with each other on bandwidth of 2 Mbps. During the communication between the two hosts, Host A performs the handover between MAR A and MAR B consecutively, and during this we measured the handover delay. The measured handover delay was established as the time required for the communication with Host 2 to be recovered after initiating the handover at Host 1.

4.4. Handover delay results

Handover experiments were conducted for cases in which the MN communicates with two different CNs, one which is physically close and the other which is physically remote. In both experiments, the MN moved only within the lowest DHT region. Because MAR A and MAR B participate in the same lowest DHT, the query was performed in a single overlay hop. The experiments were performed 100 times each, and the handover delay and components that comprise the handover delay were measured.

In order to measure the handover delay when there was a CN which was physically distant, we used an additional CN operating in California. We had the CN in California run the iPerf application to send packets to the MN in the testbed. However, because the MAR could not be physically located in California, the iPerf packet generated from the CN arrived at MAR C, causing MAR C to undertake packet processing. Therefore, the cache update message was designed to use 155 ms, i.e., the RTT between the CN and MN, to simulate MAR C being physically far from MAR A and MAR B. Table 2 shows the experimental results.

The $T_{L2\_Association}$ and $T_{DHCP\_Process}$ values differ according to the wireless network technology and the host configuration; hence, there were no differences between the two experiments. T_{HDHT_Query} was found to be 16.7 ms on average

Table 2. Handover delays

MN at Daejeon	Avg. (ms)	S.D. (ms)
$T_{L2\ Association}$	65.7	4
$T_{DHCP\ Process}$	11.6	4
$T_{HDHT\ Query}$	16.7	4.1
RTT_{CN-MN} (CN at Daejeon)	11.7	-
RTT_{CN-MN} (CN at California)	155	-

because the MN moves within the AS area. In other words, we confirmed that the handover delay depends on RTT_{CN-MN}. This result indicates that the entry-level specification can reduce the HDHT query delay to a small fixed value. This enables outstanding handover performance because the queries initiated from the MARs in the movement range of the MN can be made in a short time.

For example, consider the case where an entry pertaining to the MN is stored in levels '2' and '3' when the MN initially connects to a MAR belonging to DHT_{11}^{3} and then moves to a MAR belonging to DHT_{12}^{3}, as shown in Figure 2. When the entry is stored at level '2', the average query time was 29.9 ms. Therefore, the handover delay can be increased by approximately 13.4 ms. This value has no significant effect on the total handover time. However, when the entry was stored at level '3', the query time increased to an average of 121 ms. In this case, because the handover delay increased by approximately 104.4 ms, the effect is significant. For this reason, it is necessary to determine the HDHT level of the entry while considering the movement range of the MN.

4.5. Comparison with a state-of-the-art design

In this section, we describe the results of a logical performance comparison between a state-of-the-art design, in this case MDHT [27], and MoHiD.

First, in terms of the resolution delay, MDHT must propagate the query to the other lowest DHT through the top DHT. This is the case when the level of the entry is set to '3' in MoHiD. On the other hand, because MoHiD can specify the level at which the entry is to be stored, we can limit the number of logical hops to make the queries faster. When using the MDHT approach, the query delay always corresponds to the case in which the entry level is '3' in MoHiD. In MoHiD, if the entry level is set to '1' or '2', the number of logical hops can be smaller than that at the entry level of '3'; thus, the desired entry can be obtained in less time.

Second, in terms of scalability, because only the lowest DHT has the locator information of the nodes, the upper DHTs must retain the information pertaining to the position of the lowest DHT. Therefore, the MDHT scheme requires that the top-level DHT maintain the mapping information for all nodes in the network. MoHiD, on the other hand, is advantageous because the DHT does not need to maintain information about all nodes. The upper DHT can

determine which level of DHT to deliver when the query is forwarded.

Finally, the MDHT scheme has not been tested on the handover of the MNs. If mobility is supported based on the MDHT method, the CN must inquire about the new location after recognizing the movement of the MN. This results in additional latency in the handover support. As MoHiD immediately sends an update message to the MARs requiring the location information of the MN as soon as the MN moves, the queries are not generated from the MARs and the handover delay can be reduced. In general, the performance of DHT-based mobility platforms is greatly affected by DHT query delay, as shown in Section 3.4. Therefore, the DHT-based mobility platform has a common problem in that the query delay is increased when it is applied on the global scale because it does not consider the movement range of the MNs. On the other hand, MoHiD solves this problem because it determines the HDHT level in advance considering the range of mobility of the MNs.

5. Conclusion and future work

In this paper, we proposed the MoHiD platform. This platform includes the HDHT such that the level at which the entry is stored can be specified and the MAR that supports host mobility based on the HDHT. MoHiD operates as a full-DMM scheme. This resolves the scalability, handover performance, and single-point-of-failure issues, which continue to be problems associated with centralized mobility support methods. We used a commercial cloud service to measure the query delay of the HDHT on a global scale, and we constructed a testbed to conduct handover experiments. As future work, we plan to adapt a network controller that monitors the movement range of the MN and that automatically adjusts the HDHT level of an entry.

6. Acknowledgement

This research was part of the project entitled “SMART-Navigation project,” funded by the Ministry of Oceans and Fisheries of Korea. This work was also supported by a grant from the National Research Foundation of Korea (NRF) funded by the Korean government (MSIP) (No. 2017R1A2B4005865).

7. References

[1] Cisco Internet Business Solutions Group (IBSG), The Internet of Things: How the Next Evolution of the Internet Is Changing Everything, Apr. 2011.
[2] Gartner, By 2020, a Quarter Billion Connected Vehicles Will Enable New In-Vehicle Services and Automated Driving Capabilities, Jan. 2015.
[3] Machina Research, Global M2M Market to Grow to 27 Billion Devices, Generating USD 1.6 Trillion in Revenue in 2024, Jun. 2015.
[4] 3GPP, Evolved General Packet Radio Service (GPRS) Tunnelling Protocol for Control plane (GTPv2-C), 3rd Generation Partnership Project (3GPP), TS 29.274, September 2011.
[5] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, Proxy Mobile IPv6, RFC 5213, August 2008.
[6] H. Chan, et al., Requirements for Distributed Mobility Management, IETF RFC 7333, Apr. 2014.
[7] D. Liu, and S. Pierrick, Distributed Mobility Management: Current practices and Gap Analysis, IETF RFC 7429, 2015.
[8] C. Perkins, IP Mobility Support for IPv4, Revised, IETF RFC 5944, Nov. 2010.
[9] C. Perkins, D. Johnson, and J. Arkko, Mobility Support for IPv6, IETF RFC 6275, Jul. 2011.
[10] C. Dannewitz, D. Matteo and V. Vinicio, Hierarchical DHT-based Name Resolution for Information-Centric Networks, Computer Communications 36.7 (2013): 736-749.
[11] L. Garces-Erice, et al., Hierarchical Peer-to-Peer systems, Parallel Processing Letters 13.04 (2003): 643-657.
[12] Z. Xu, R. Min, and Y. Hu, HIERAS: a DHT based Hierarchical P2P Routing Algorithm, IEEE Parallel Processing, 2003.
[13] P. Ganesan, K. Gummadi, and H. P. Garcia-Molina, Canon in G major: Designing DHTs with Hierarchical Structure, IEEE ICDCS, 2014.
[14] P. Fonseca, et al., Full-information Lookups for Peer-to-Peer Overlays, IEEE Transactions on Parallel and Distributed Systems 20.9 (2009): 1339-1351.
[15] P. Maymounkov, and D. Mazieres, Kademlia: A Peer-to-Peer Information System based on the XOR Metric, International Workshop on Peer-to-Peer Systems. Springer Berlin Heidelberg, 2002.
[16] P. Wang, et al., Towards Locality-aware DHT for Fast Mapping Service in Future Internet, Computer Communications 66 (2015): 14-24.
[17] F. Giust, L. Cominardi, and C. Bernardos, Distributed Mobility Management for Future 5G Networks: Overview and Analysis of Existing Approaches, IEEE Communications Magazine, 53(1): 142-149, 2015.
[18] C. J. Bernardos, A. Oliva, and F. Giust, A PMIPv6-Based Solution for Distributed Mobility Management, IETF draft, July. 2014.
[19] I. Stoica, et al., Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, ACM SIGCOMM Computer Communication Review 31.4 (2001): 149-160.
[20] S. Ratnasamy, et al., A Scalable Content-addressable Network, Vol. 31. No. 4. ACM, 2001.
[21] R. Cox, A. Muthitacharoen, and R. T. Morris, Serving DNS using a Peer-to-Peer Lookup Service, International Workshop on Peer-To-Peer Systems. Springer Berlin Heidelberg, 2002.
[22] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, The Locator/ID Separation Protocol (LISP), IETF RFC 6830, Jan. 2013.
[23] D. Han, et al., XIA: Efficient Support for Evolvable Internetworking, USENIX NSDI, 2012.
[24] I. Seskar, et al., Mobilityfirst Future Internet Architecture Project, Proceedings of the 7th Asian Internet Engineering Conference. ACM, 2011.
[25] H. Luo, Y. Qin, and Z. Hongke, A DHT-based Identifier-to-Locator Mapping Approach for a Scalable Internet, IEEE Transactions on Parallel and Distributed Systems 20.12 (2009): 1790-1802.
[26] T. Vu, et al., Dmap: A Shared Hosting Scheme for Dynamic Identifier to Locator Mappings in the Global Internet, IEEE ICDCS, 2012.
[27] M. D'Ambrosio, et al., MDHT: A Hierarchical Name Resolution Service for Information-centric Networks, Proceedings of the ACM SIGCOMM workshop on Information-centric networking. ACM, 2011.
[28] V. Ramasubramanian, and E. G. Sirer, The design and implementation of a next generation name service for the internet, ACM SIGCOMM Computer Communication Review, 34(4), pp. 331-342, 2004.
[29] NIST, Descriptions of SHA-256, SHA-384, and SHA-512, May. 2001.
[30] S. Han, et al., Mobility of Everything (MoE): An Integrated and Distributed Mobility Management, IEEE ICCCN, 2017.
[31] Open vSwitch, http://openvswitch.org/.
[32] Mininet - An Instant Virtual Network on your Laptop (or other PC), http://mininet.org/.
[33] Amazon Elastic Compute (EC2), http://aws.amazon. com/ec2/.
[34] iPerf - The TCP/UDP Bandwidth Measurement Tool, https://iperf.fr